

Stochastic dominance and the bijective ratio of online algorithms*

Spyros Angelopoulos¹, Marc P. Renault², and Pascal Schweitzer³

¹CNRS and Université Pierre et Marie Curie, Paris, France, spyros.angelopoulos@lip6.fr

²CNRS and Université Paris Diderot, Paris, France, mrenault@liafa.univ-paris-diderot.fr

³RTWH Aachen University, Aachen, Germany, schweitzer@informatik.rwth-aachen.de

Abstract

Stochastic dominance is a technique for evaluating the performance of online algorithms that provides an intuitive, yet powerful stochastic order between the compared algorithms. Accordingly this holds for *bijective analysis*, which can be interpreted as stochastic dominance assuming the uniform distribution over requests. These techniques have been applied in problems such as paging, list update, bin coloring, routing in array mesh networks, and in connection with Bloom filters, and have provided a clear separation between algorithms whose performance varies significantly in practice. However, despite their appealing properties, there are situations in which they are not readily applicable. This is due to the fact that they stipulate a stringent relation between the compared algorithms that may be either too difficult to establish analytically, or worse, may not even exist.

In this paper we propose remedies to both of these shortcomings. First, we establish sufficient conditions that allow us to prove the bijective optimality of a certain class of algorithms for a wide range of problems; we demonstrate this approach in the context of well-studied online problems such as weighted paging, reordering buffer management, and 2-server on the circle. Second, to account for situations in which two algorithms are incomparable or there is no clear optimum, we introduce the *bijective ratio* as a natural extension of (exact) bijective analysis. Our definition readily generalizes to stochastic dominance. This renders the concept of bijective analysis (and that of stochastic dominance) applicable to all online problems, is a broad generalization of the Max/Max ratio due to Ben-David and Borodin, and allows for the incorporation of other useful techniques such as amortized analysis. We demonstrate the applicability of the bijective ratio to one of the fundamental online problems, namely the continuous k-server problem on metrics such as the line, the circle, and the star. Among other results, we show that the greedy algorithm attains bijective ratios of $O(k)$ consistently across these metrics. These results confirm extensive previous studies that gave evidence of the efficiency of this algorithm on said metrics in practice, which, however, is not reflected in competitive analysis.

*This work was supported by project ANR-11-BS02-0015 “New Techniques in Online Computation” (NeTOC). The second author is supported in part by the European Research Council under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 648032).

1 Introduction

Competitive analysis provides a simple yet effective framework for evaluating the performance of online algorithms. Given a cost-minimization problem, the *competitive ratio* of the online algorithm A is defined as $\sup_{\sigma} \frac{A(\sigma)}{\text{OPT}(\sigma)}$, where $A(\sigma)$ and $\text{OPT}(\sigma)$ denote the cost of A and the optimal cost on a request sequence σ , respectively. This concept of comparing the worst-case performance of an online algorithm (with no advance knowledge of the sequence) to an optimal solution (with full access to the sequence) was first used by Graham in 1966 [24] to analyze algorithms for the job shop scheduling problem. Following the seminal work of Sleator and Tarjan in 1985 [44], competitive analysis became the standard yardstick in the evaluation of online algorithms, and it has been instrumental in shaping online computing into a well-established field of theoretical computer science. Overall, competitive analysis is broadly applicable and gives valuable insight into the performance of online algorithms.

Notwithstanding the undeniable success of competitive analysis, certain drawbacks have long been known. Most notably, due to its pessimistic (i.e., worst-case) evaluation of algorithms, it often fails to distinguish between algorithms for which experimental evidence (or even plain intuition) would show significant differences in terms of performance. One definitive illustration of this undesirable situation is the well-known *paging problem*; here, paging strategies that are very efficient in practice, such as Least-Recently-Used have the same competitive ratio as extremely naive and costly strategies, such as Flush-When-Full [44]. Generally, competitive analysis is particularly meaningful when the obtained ratios are small; however, when the ratios are large (and even more so, in the case of an unbounded ratio) it risks no longer reflecting what is observed in practice. Such disconnects between the empirical and the theoretical performance evaluation have motivated a substantial line of research on measures alternative to the competitive ratio. Some of the known approaches are: the *Max-Max ratio* [11]; the *diffuse adversary model* [32, 48, 49]; *loose competitiveness* [47, 50]; the *random order ratio* [30]; the *relative worst-order ratio* [15, 14]; the *accommodation function model* [17]; and *stochastic dominance* as well as *bijective and average analysis* [28, 18, 19, 42, 36, 34, 3, 25, 4, 5]. We refer the reader to the surveys [23, 27] for an in-depth discussion of such techniques.

Of particular interest in this work is (first order) *stochastic dominance* which defines a partial order on random variables. A random variable X is stochastically dominated by a random variable Y if, for all $c \in \mathbb{R}$, we have $\Pr[X \leq c] \geq \Pr[Y \leq c]$. Clearly, if X is stochastically dominated by Y , then $\mathbb{E}(X) \leq \mathbb{E}(Y)$; however a much stronger conclusion can be drawn, namely that, for the cumulative distribution functions of the distributions from which X and Y are drawn, denoted by F and G respectively, $F(c) \geq G(c)$ for all c . That is, informally, F has more probability mass towards lower values than G . Moreover, it can be shown that $\mathbb{E}(h(X)) \leq \mathbb{E}(h(Y))$ for every increasing function h . If we think of h as a utility function, then stochastic dominance provides a kind of *unanimity rule* which informally states that X should be preferred to Y under any monotone utility function (assuming these variables denote costs). For this reason, stochastic dominance has been very useful in the context of decision theory and microeconomics, with applications varying from portfolio selection to measuring income inequality in society. For a comprehensive discussion, see Chapters 4 and 5 in the textbook [45].

Hiller and Vredeveld [25] applied the concept of stochastic dominance in the context of online computing. More precisely, an algorithm A is *stochastically no worse* than an algorithm B with respect to a given distribution over the request sequences if the random variable corresponding to the cost of A is stochastically dominated by that of B . In particular, assuming the uniform

distribution over all request sequences of a given size, stochastic dominance is equivalent to *bijective analysis* [27]. This latter notion was first introduced in [3] in the context of the paging problem and was shown to be consistent with some natural, “to-be-expected” properties of efficient online algorithms (e.g., the effect of locality of reference as well as lookahead) which competitive analysis fails to yield. For a further discussion of the appealing aspects of bijective analysis, see [3, 5].

Definition 1 ([3]). *Let \mathcal{I}_n denote the set of all request sequences of size n . The online algorithm A is no worse than the online algorithm B on inputs of size n according to bijective analysis if there exists a bijection $\pi : \mathcal{I}_n \rightarrow \mathcal{I}_n$ satisfying $A(\sigma) \leq B(\pi(\sigma))$ for each $\sigma \in \mathcal{I}_n$. Moreover, A is bijectively optimal if the above holds for all online algorithms B .*

The bijective ratio of online algorithms. Despite the appealing properties of bijective analysis, its biggest deficiency is a rather serious one: given two online algorithms it may be very difficult to compare them, in that it may be very hard to prove analytically the existence of the required bijection; even worse, such a bijection may not even exist. Thus, this analysis technique may deem algorithms incomparable across a wide variety of problems, and, in this sense, it does not give rise to a real performance measure. This drawback implies that bijective analysis (and by extension, stochastic dominance more generally) lacks the most desirable property of the competitive ratio; namely the amenability of any given online problem to analysis. Such an observation could also help explain why these techniques did not become as popular as competitive analysis, even though the fundamentals and some limited applications can be traced to work contemporary of competitive analysis. Calderbank et al. [18] point towards such difficulties when observing in the context of the k -server problem that “The prospects for successful analysis would seem to be better for the circle. However, even in this case optimization questions may well be intractable since rules of the simplicity of the [greedy] rule are unlikely to be optimal” (see also the discussion in Section 2). For this reason, [3] introduced a substantially weaker technique termed *average analysis* that compares the average cost of two algorithms over requests of the same length. In particular, we say that A is *no worse* than B on inputs of size n according to average analysis if $\sum_{\sigma \in \mathcal{I}_n} A(\sigma) \leq \sum_{\sigma \in \mathcal{I}_n} B(\sigma)$. Note that, if A is no worse than B according to bijective analysis, then the same relation holds for average analysis; however the opposite is not necessarily true.

In this paper, we propose (and apply) an extension of bijective analysis that makes the technique applicable to any given online problem: this extension gives rise to a performance measure which we call the *bijective ratio*. This is equivalent to an approximate stochastic dominance under a uniform distribution and can be readily generalized to the *stochastic dominance ratio* for any distribution.

Definition 2. *Given an online algorithm A and an algorithm B , and $n \in \mathbb{N}^+$, we say that the bijective ratio of A against B is at most ρ if there exists a bijection $\pi : \mathcal{I}_n \rightarrow \mathcal{I}_n$ satisfying $A(\sigma) \leq \rho \cdot B(\pi(\sigma))$ for all $n \geq n_0$. We denote this by $A \preceq_b \rho \cdot B$. The bijective ratio of an online algorithm A is at most ρ if, for every algorithm B , the bijective ratio of A against B is at most ρ . The bijective ratio of an online cost-minimization problem is the minimum ρ for which there exists an online algorithm with bijective ratio at most ρ .*

We note that, in Definition 2, one may allow B to be either an online, or an offline algorithm (and in particular, the offline optimum). We can thus distinguish between the bijective ratio of an online algorithm *against online or offline algorithms*. We clarify that unless explicitly specified, “the bijective ratio of an algorithm A ” assumes a comparison against the offline optimum.

The above distinction is motivated similarly to the Max/Max ratio introduced by Ben-David and Borodin [11], which is defined as the ratio of the maximum-cost sequence for algorithm A over the maximum-cost sequence for algorithm B (which may be online or offline), for a given sequence length. We emphasize that the bijective ratio is a strong generalization of the Max/Max ratio; namely it implies that the cost of the i -th most expensive sequence of A is at most ρ times the cost of the i -th most expensive sequence of B for all i , and not just for the most expensive sequences of A and B .

Definition 2 is a natural extension of bijective optimality in the spirit of measures such as the competitive and the approximation ratio. It also upholds the essential aspect of bijective analysis in that every sequence for which A incurs a certain cost can be bijectively mapped to a sequence on which B is at most ρ times as costly. Furthermore, a bijective ratio of ρ implies that the average-cost ratio of the two algorithms is at most ρ , but also the far stronger conclusion that the contribution of sequences to the average costs of the two algorithms can be attributed in a local manner, as argued above. This aspect extends to any stochastic dominance ratio for any distribution. Both properties are desired extensions of bijective optimality, in the sense that they provide a much stronger comparison than the one induced by average-case analysis.

Last, we note that in the above definitions, the performance ratios are *strict*; however, as with the competitive ratio, one can easily define *asymptotic* ratios. For instance, the asymptotic ratio of A against B is at most ρ if there exists a constant c such that $A(\sigma) \leq \rho \cdot B(\pi(\sigma)) + c$ for all $n \geq n_0$.

Contribution. Our main objective is to expand the applicability of stochastic dominance and, more specifically, bijective analysis. We accomplish this in two ways: first, by giving general, sufficient conditions for bijectively optimal algorithms; second, by applying the measure of bijective ratio to one of the canonical online problems, namely the k -server problem, as a case study.

We begin our study of bijective analysis in Section 3, in which we extend the techniques applied in [5] so as to prove the bijective optimality of certain types of greedy algorithms for a much wider class of problems than paging and list update. In particular, we identify some essential conditions under which a certain subclass of greedy-like algorithms (as formally defined in [13]) are optimal. We then apply this general framework to the *2-server problem on the continuous circle*, the *weighted paging* problem, and the *reordering buffer management* problem. The above are all widely studied problems in online computing. In particular, the result for the 2-server problem on the continuous circle improves on the result of Calderbank et al. [18] which holds for the average case only. We also note that Anagnostopoulos et al. [2] studied the steady state of a stochastic version of the k -server problem on the circle, and reproved the optimality, in the average case, of the greedy algorithm when $k = 2$.

Our second contribution addresses the situation in which, according to stochastic dominance or bijective analysis, optimal algorithms may not necessarily exist. More precisely, we demonstrate the applicability of the bijective ratio in the analysis of the continuous k -server problem on the line, circle, and star metrics. Our main focus is on the performance of the greedy algorithm which is motivated by several factors. First, and most importantly, there is ample experimental evidence that in practice the greedy algorithm performs well in several settings [18, 10, 40, 41]. However, these results are in stark contrast with competitive analysis since the greedy algorithm has an unbounded competitive ratio even on the line. As noted in [40], “the [experimental] results demonstrate that [Work Function Algorithm (WFA)] performs similarly or only slightly better than a simple heuristic such as the greedy algorithm, although according to theory it should perform much better”. In

this sense, there is a big disconnect between theoretical and practical behaviour which, perhaps surprisingly, has not received as much attention from the theoretical computer science community as other problems such as the paging problem. Our results demonstrate that bijective analysis can help bridge this gap. More precisely, we show that the greedy algorithm has bijective ratio $O(k)$ in the considered metric spaces. Note that, for the k -server problem on the circle, we obtain a bijective ratio of k , while the best-known competitive ratio is $2k - 1$ by the analysis of the WFA [12].

Another appealing property of the greedy algorithm for the k -server problem, which is also true for the other online problems we study, is that they are among the simplest *memoryless* algorithms one can devise. Memoryless algorithms are very desirable, in general, and are particularly important for paging problems [39], and controlling disk heads [18]. In the context of the k -server problem, in particular, it is known that WFA is prohibitive in practice as it requires a full history of the requests and is much more complicated to implement than the simple greedy algorithm [10, 40, 41]. Our result on the bijective optimality of a greedy policy concerning the weighted paging problem is of note given that [20] showed that no deterministic memoryless algorithm has bounded competitive ratio for this problem.

In Section 4, we first show that the greedy algorithm (denoted by GREEDY) is not an optimal online algorithm for 2-server on the line, even for average-case analysis (which implies the same result for bijective analysis). This improves on a result of Calderbank et al. [18] that showed that there exists a *semi-online* algorithm (that knows the length of the sequence) that outperforms GREEDY only on the last two requests. We also show that no online algorithm has a strict bijective ratio better than 2 for this problem. This immediately raises the question: How good (or bad) is GREEDY? We address this question by showing that GREEDY has a strict bijective ratio of at most k and $2k$ for the circle and the line, respectively; for the line, we also obtain an asymptotic bijective ratio at most $4k/3$. This analysis is almost tight, since we show that the asymptotic bijective ratio of GREEDY is at least $k/3 - \epsilon$ and $k/2 - \epsilon$, for the circle and the line, respectively. We also consider the algorithm K-CENTER [11], which anchors its servers at k points of the metric so as to minimize the maximum distance of any point in the metric to a server; it then serves each request by moving the closest server which subsequently returns to its anchor position. In contrast to the results for GREEDY, K-CENTER has an asymptotic bijective ratio of 2 for the line and the circle which generalizes the known bound on the Max/Max ratio of this algorithm [11]. In terms of a direct comparison of online algorithms, we obtain that GREEDY has a bijective ratio of at most $2k/3$ against K-CENTER. It is worth mentioning that our results expand the work of Boyar et al. [16] who showed that GREEDY is bijectively optimal for the 2-server problem on a very simple, albeit discrete metric consisting of three colinear points (termed the *baby server problem*).

Last, in Section 5 we consider the continuous k -server problem on star-like metrics. Here, we show that the bijective ratio of the greedy algorithm is at most $4k$. On the negative side, we show that K-CENTER is unbounded for such metrics. This raises an interesting contrast between the bijective ratio and the Max/Max ratio: while K-CENTER has Max/Max ratio at most $2k$ for k -server on any bounded metric space, when considering the bijective ratio (which, as noted earlier, generalizes the Max/Max ratio), this algorithm becomes very inefficient.

In terms of techniques, the transition from exact to approximate bijective analysis necessitates a new approach that combines bijective analysis and amortization arguments. In particular, we note that all previous work that establishes the bijective optimality of a given algorithm [3, 4, 5, 16] is based on inductive arguments which do not immediately carry over to the bijective ratio. For instance [5] crucially exploits the fact that for $\rho = 1$, if $A \preceq_b \rho \cdot B$ and $B \preceq_b \rho \cdot C$, then $A \preceq_b C$. This

obviously only holds for $\rho = 1$, i.e., for optimality. We thus follow a different approach that is based on a *decoupling* of the costs incurred by the compared algorithms (stated formally in Lemmas 17 and 18) by formulating two desirable properties: the first property captures the “local” efficiency of the greedy algorithm (but also potentially other good algorithms), while the second property allows us to define best and worst server configurations (or approximations thereof) that provide insights into the choice of the appropriate bijection. Combining these properties yields the desired results. For line and star metrics, in particular, we resort to amortized analysis using explicit potential functions which is the first example of a combination of bijective and amortized analysis.

We conclude this section with two observations. First, we use the bijective ratio both to compare algorithms against the optimal offline algorithm (similar to the competitive ratio) and to directly and indirectly compare online algorithms. As an example of indirect comparison, Theorem 24 implies that K-CENTER has a bijective ratio of at most 2 against GREEDY, which, in combination with Theorem 16 implies that GREEDY has a bijective ratio of $\Omega(k)$ against K-CENTER for the line and the circle. The latter is asymptotically tight due to a direct-comparison result (stated in Theorem 25). Second, while our focus is mainly on the greedy algorithm for reasons argued earlier, our techniques (in particular the decoupling Lemmas 17 and 18) are not tied to GREEDY or K-CENTER, and are potentially applicable to a wider class of algorithms.

2 Related work and preliminaries

Related work. Stochastic dominance (cf. [43, 37, 45, 27]) is a widely established concept in decision theory. Optimal algorithms, assuming certain pertinent distributions, have been identified for various online problems such as the paging problem [3, 5, 26], the list update problem [4, 5], routing in array mesh networks [36], bin colouring [25] and in the online construction of Bloom filters [34]. The first application of stochastic dominance for the analysis of online algorithms can be traced back to [28, 42] in the context of the *two-headed disk* problem. This problem is related to the k -server problem but with a different cost function. Given k mobile servers on a metric space, request appear on the points of the metric space and the goal is to minimize the distance travelled to serve these requests. During the time a request is being served, the other servers can re-position themselves at no cost. This renders the decision of which server to use trivial (it will always be the closer server) and puts focus on the question of where to place the other servers. Hofri showed that the natural greedy algorithm for this problem on the line is optimal in average and conjectured that it is stochastically dominated by every other algorithm under a uniform distribution [28] which was proven by Seshadri and Rotem [42].

The k -server problem, originally proposed by Manasse et al. [35], involves k mobile servers over a metric space. Upon a request to a node, a server must be moved to it. The incurred cost is the overall distance traversed by the servers. The k -server problem generalizes the paging problem and has motivated an outstanding body of research (see the surveys [22] and [31]). In general metric spaces, the Work Function Algorithm (WFA) of Koutsoupias and Papadimitriou is $(2k - 1)$ -competitive [33]; the best-known lower bound on the deterministic competitive ratio is k [35]. WFA is also known to be k -competitive on the line [9] as well as for two servers in general metric spaces [22], and it is the best known algorithm for the circle [12]. Chrobak and Larmore showed that the algorithm Double Coverage, which moves certain servers at the same speed in the direction of the request until a server reaches the requested point, is k -competitive for the tree metric [21]. Calderbank et al. studied the 2-server problem on the line and circle [18], and the n -dimensional

sphere [19]. They focused on the average case and, in particular, calculated the expected cost of GREEDY on the circle. Moreover, [18] presents experimental data that show that GREEDY is relatively close in performance to the offline optimal algorithm on the line. Similar experiments, for a variety of metric spaces and algorithms, including GREEDY, are presented in [10, 40, 41]. In a related work, Anagnostopoulos et al. [2] studied the steady-state distribution of GREEDY for the k -server problem on the circle.

Boyar et al. [16] provided a systematic study of several measures for a simple version of the k -server problem, namely, the two server problem on three colinear points. In particular, they showed that GREEDY is bijectively optimal. Concerning the Max/Max ratio, [11] showed that the algorithm K-CENTER is asymptotically optimal up to a factor of 2 among all online algorithms and up to a factor of $2k$ from the optimal offline algorithm.

Preliminaries. We denote by σ a sequence of requests, and by \mathcal{I}_n the set of all request sequences of size n . Following [5], we denote by $\sigma[i, j]$ the subsequence $\sigma[i] \dots \sigma[j]$. We also use sometimes σ_i to refer to the i -th request of σ , namely $\sigma[i]$. For the k -server problem, we denote the distance between two points x, y by $d(x, y)$. Unless otherwise noted, we assume that both the line and the circle have unit lengths.

Since, for continuous metrics, \mathcal{I}_n is infinite, one needs to be careful about the allowable bijections. We model the continuous k -server problem using discrete metrics in which nodes are placed in an equispaced manner; as the number of nodes approaches infinity, this model provides a satisfactory approximation of the continuous problem. For instance, we approximate the continuous line (resp. circle) by a path (resp. cycle) in which vertices are uniformly spaced, i.e., all edges have the same length which may be arbitrarily close to zero. However, we note that the techniques we use in this paper are applicable even for the formal definition of the continuous problem, i.e., even when the set of all request sequence of size n is infinite. However, in this case one needs to be careful about the allowable bijections. For instance, we should not allow bijections that map the unit line to segments of measure strictly smaller than one. For this reason, we restrict the allowable bijections to *interval exchange transformations* [29]. These transformations induce bijections of the continuous space $[0, 1]$ to itself that preserve the Lebesgue measure. Note that an interval exchange transformation is continuous with the exception of a finite number of points. In particular, we apply such transformations when constructing the bijection request-by-request.

Given an online algorithm A , we say that the *configuration* of A after serving any sequence of requests σ is the state of the algorithm immediately after serving σ , where the notion of “state” will be implicit in the definition of the online problem. For example, in the k -server problem, this would be the position of the servers in the metric space.

3 A sufficient condition for optimality of greedy-like algorithms

In this section, we show how the techniques of [5] can be applied in a variety of online problems, so as to prove that certain greedy algorithms are bijectively optimal among all online algorithms. To this end, we first need a criterion that establishes, in a formal manner, the greedy characteristic. More precisely, consider an online algorithm that must serve request σ_i after having served the sequence $\sigma[1, i - 1]$. We say that an algorithm is *greedy-like* if it serves each request σ_i in a way that minimizes the cost objective, assuming this request σ_i is the final request. This definition

is motivated by a similar characterization of “greediness” in the context of *priority algorithms* as defined by Borodin et al. [13].

Naturally, not all greedy-like algorithms are expected to be bijectively optimal. For instance, for the classic paging problem, all lazy algorithms are greedy-like, however, as shown in [3, 5], assuming locality of reference, only LRU is optimal. Therefore, one needs to choose a “good” algorithm in this class of greedy-like algorithms. Let G denote such a greedy-like algorithm. We say that A is G -like on σ_i if, after serving $\sigma[1, i-1]$, A serves request σ_i as G would. Note that the G -like notion cannot be characterized in general for all online problems. It needs to be defined for specific problems and specific greedy-like algorithms (e.g., the definition of an “LRU-like” algorithm in [5]). Given sequences over \mathcal{I}_n , Algorithm A is G -like on the suffix $[j, n]$ if A serves all requests $\sigma_j \dots \sigma_n$ in a G -like manner. The following definition formally describes algorithms for which the G -like decision can be moved “one step earlier” without affecting performance with respect to bijective analysis.

Definition 3. Suppose that A is an online algorithm over sequences in \mathcal{I}_n such that A is G -like on the suffix $[j+1, n]$. We say that A is G -like extendable on j if there exists a bijection $\pi : \mathcal{I}_n \rightarrow \mathcal{I}_n$ and an online algorithm B with the following properties.

- For every $\sigma \in \mathcal{I}_n$, B makes the same decisions as A on the first $j-1$ requests of σ .
- For every $\sigma \in \mathcal{I}_n$, B is G -like on σ_j .
- $\pi(\sigma)[1, j] = \sigma[1, j]$ and $B(\pi(\sigma)) \leq A(\sigma)$.

Informally, A is G -like extendable if it can be transformed to another algorithm B that is “closer” to the statement of a G -like algorithm and is not inferior to A according to bijective analysis. We note that Definition 3 is motivated by the statement of Lemma 3.4 in [5]; in contrast to the latter, it applies not only to paging (and the LRU algorithm) but to all online problems for which there is a well-defined G algorithm with the above properties (and in particular, is greedy-like). This definition is instrumental in proving the optimality of G ; in particular, we obtain the following theorem. The proof follows along the lines of the proof of Lemma 3.7 and Theorem 3.8 in [5].

Theorem 4. If every online algorithm A (over requests in \mathcal{I}_n) that is G -like on the suffix $[j+1, n]$ is also G -like extendable on j , for all $1 \leq j \leq n$, then G is optimal.

Proof. Consider an arbitrary algorithm ALG and any request sequence $\sigma \in \mathcal{I}_n$. We will show that there exists a bijection $\pi : \mathcal{I}_n \rightarrow \mathcal{I}_n$ such that $G(\sigma) \leq \text{ALG}(\pi(\sigma))$.

Fix an arbitrary $\sigma \in \mathcal{I}_n$, we will show by reverse induction on the requests that the theorem holds. More precisely, let \mathbb{C}_i be the set of all algorithms that are G -like on $\sigma[i, n]$ and serve $\sigma[1, i-1]$ exactly as ALG . By reverse induction on the indexes of σ , we show that, for every algorithm ALG , there exists an algorithm $C_i \in \mathbb{C}_i$ and a bijection μ_i such that $C_i(\sigma) \leq \text{ALG}(\mu_i(\sigma))$ and $\mu_i(\sigma)[1, i] = \sigma[1, i]$.

For the last request, define μ_n to be the identity function, and define C_n to serve $\sigma[1, n-1]$ exactly as ALG and to serve σ_n in a G -like manner. The claim follows immediately from the fact that G is greedy-like.

Consider the inductive step from $i+1$ to i . From the induction hypothesis, there exists an algorithm $C_{i+1} \in \mathbb{C}_{i+1}$ such that

$$C_{i+1}(\sigma) \leq \text{ALG}(\mu_{i+1}(\sigma)) . \tag{1}$$

By the theorem statement, C_{i+1} is G -like extendable on i and, by Definition 3, we have an algorithm B_i and a bijection π_i such that

$$B_i(\sigma) \leq C_{i+1}(\pi_i^{-1}(\sigma)) \leq \text{ALG}(\mu_{i+1}(\pi_i^{-1}(\sigma))) , \quad (2)$$

where the last inequality follows from (1). Note that $\pi_i^{-1}(\sigma)[1, i] = \pi_i(\sigma)[1, i] = \sigma[1, i]$

By applying the induction hypothesis on algorithm B_i , there exist an algorithm $C_i \in \mathbb{C}_{i+1}$ and a bijection μ'_{i+1} such that

$$C_i(\sigma) \leq B_i(\mu'_{i+1}(\sigma)) \leq \text{ALG}(\mu_{i+1}(\pi_i^{-1}(\mu'_{i+1}(\sigma)))) ,$$

where the last inequality follows from (2).

Since $C_i \in \mathbb{C}_{i+1}$, C_i is G -like on $\sigma[i+1, n]$. Moreover, as $C_i \in \mathbb{C}_{i+1}$ and is based on B_i , C_i is G -like on $\sigma[i]$ as it serves $\sigma[1, i]$ exactly as B_i and, by Definition 3, B_i is G -like on $\sigma[i]$. It follows then that C_i serves $\sigma[1, i-1]$ exactly as ALG and is G -like on $\sigma[i, n]$. Hence, $C_i \in \mathbb{C}_i$. Define $\mu_i := \mu_{i+1} \circ \pi_i^{-1} \circ \mu'_{i+1}$. Note that $\mu_i(\sigma)[1, i] = \mu_{i+1}(\pi_i^{-1}(\mu'_{i+1}(\sigma[1, i]))) = \sigma[1, i]$ and the inductive step follows.

After the induction, there exists an algorithm $C_1 \in \mathbb{C}_1$. Algorithm C_1 is G -like on $\sigma[1, n]$ and, therefore, $G(\sigma) = C_1(\sigma) \leq \text{ALG}(\pi(\sigma))$, where $\pi := \mu_1$. \square

We will demonstrate the applicability of this framework by showing optimality of greedy-like online algorithms for three well-known online problems: the 2-server problem on the circle, the weighted paging problem and the reordering buffer management problem.

3.1 The 2-server problem on the continuous circle

We begin with the 2-server problem on the continuous circle. Here the candidate algorithm G is the obvious greedy algorithm (with an arbitrary tie-breaking rule) that serves a request by moving the server closer to the request, and the G -like notion is obviously well-defined.

Theorem 5. *GREEDY is optimal for 2-server on the circle.*

Proof. Let A denote any online algorithm that is G -like on the suffix $[j+1, n]$, for some $j \in [1, n]$. From Theorem 4, it suffices to prove that A is G -like extendable on j . We will show the existence of an appropriate online algorithm B and a bijection π , according to Definition 3. In particular, since the definition requires that $\pi(\sigma)[1, j] = \sigma[1, j]$, and that B makes the same decisions as A on $\sigma[1, j-1]$, we only need to define $\pi(\sigma)[j+1, n]$, as well as the decisions of B while serving the latter sequence of requests.

Consider the request σ_j : if A serves this request in a G -like manner (i.e., greedily), then the lemma holds trivially. Otherwise, note that after serving $\sigma[1, j-1]$ and $\pi(\sigma)[1, j-1]$, respectively, A and B have the same configuration. Namely, if a_1, a_2 and b_1, b_2 denote the servers for the two algorithms at this configuration, we have that $a_1 \equiv b_1$ and $a_2 \equiv b_2$. Since A does not serve σ_j greedily, we can assume, without loss of generality, that $d(a_2, \sigma_j) \geq d(a_1, \sigma_j)$ and that A serves the request using a_2 (see Figure 1 for an illustration). Let $D = d(a_2, \sigma_j) - d(a_1, \sigma_j)$, and let $\bar{\sigma}[j+1, n]$ denote the sequence which is derived from $\sigma[j+1, n]$ by *shifting* each request by $d(a_1, \sigma_j)$ in the direction opposite to the move of a_2 (in the example of Figure 1, this is done clockwise). We then define the mapping $\pi(\sigma)$ as $\pi(\sigma) = \sigma[1, j] \cdot \bar{\sigma}[j+1, n]$; it is straightforward to show that this mapping is bijective in \mathcal{I}_n .

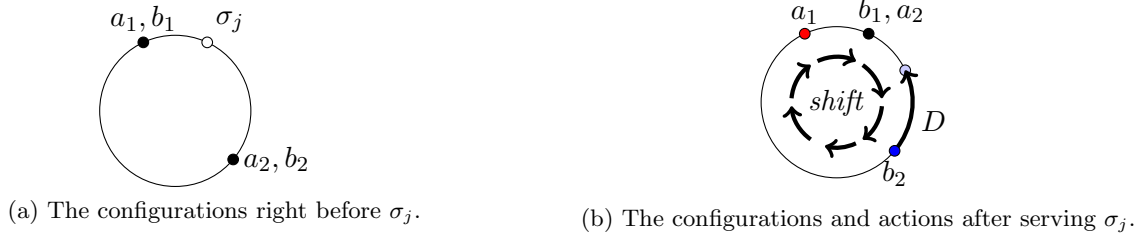


Figure 1: An illustration of the bijection that *shifts* the requests around the circle by a distance of $d(a_1, \sigma_j)$ and the first action of A after the configurations of A and B diverge on request σ_j

Next, we define the actions of algorithm B over the sequence $\pi(\sigma)[j+1, n]$. In particular, note that B serves the request $\sigma_j = \pi(\sigma_j)$ greedily; moreover we require that B subsequently moves the server b_2 by a distance equal to D (in the example of Figure 1, this is done counter-clockwise). It can be shown by induction on l , that, for all $l \in [j+1, n]$, if servers a_1, a_2 are at distance x right before A serves σ_l , then servers b_1, b_2 are at distance x before B serves $\pi(\sigma_l)$; that is, B can serve the request $\pi(\sigma_l)$ by moving one of its servers that is in the same position, relative to the shift, as the server of A that serves σ_l , and thus the two costs are identical. In conclusion, the cost of A on $\sigma[j+1, n]$ is the same as the cost of B on $\pi(\sigma[j+1, n])$, which further implies that $A(\sigma) = B(\pi(\sigma))$, which concludes the proof. \square

3.2 The weighted paging problem

Next, we consider the *weighted paging problem*. This is a generalization of the standard (uniform) paging problem, in which each page p is associated with an eviction cost c_p , and has generated an impressive body of work from the point of view of competitive analysis (see e.g., [20, 46, 8] and references therein). It is well known that the weighted paging problem for a cache of size k is equivalent to the k -server problem in a *discrete star graph*, assuming there are no requests to the center node of the star. More precisely, the star has as many edges as pages, and the weight of each edge is equal to half the eviction cost of the corresponding page; last, requests may appear on any leaf of the star.

Consider the simple greedy algorithm G that, upon a fault, evicts from the cache a page of smallest cost; clearly, this algorithm is greedy-like. The proof of the next theorem relies on Theorem 4 (as in the case of the proof of Theorem 5). However, unlike Theorem 5 (and unlike the proof of the bijective optimality of greedy/lazy algorithms for unweighted paging in [3]), the proof is technically more involved, due to the asymmetry of the cost requests (which complicates the argument for the G -like extendability of all possible online algorithms).

Theorem 6. *GREEDY is bijectively optimal for weighted paging.*

We give the proof in the framework of the k -server problem on the discrete star (which as explained, is an equivalent formulation of the weighted paging problem). Let A denote any online algorithm that is G -like on the suffix $[j+1, n]$ for some $j \in [1, n]$. From Theorem 4, it suffices to prove that A is G -like extendable on j . We will show the existence of an appropriate online algorithm B and a bijection π , according to Definition 3. In particular, since the definition requires that $\pi(\sigma)[1, j] = \sigma[1, j]$, and that B makes the same decisions as A on $\sigma[1, j-1]$, we only need to define $\pi(\sigma)[j+1, n]$, as well as the decisions of B while serving the latter sequence of requests.

Consider the request σ_j : if A serves this request greedily, then the lemma holds trivially. Otherwise, note that after serving $\sigma[1, j-1]$ and $\pi(\sigma)[1, j-1]$, respectively, A and B have the same configuration. For concreteness, let a_1, \dots, a_k , and b_1, \dots, b_k denote the configurations of A and B after serving $\sigma[1, j-1]$, and $\pi(\sigma)[1, j-1]$, respectively, with $a_i \equiv b_i$, for all $i \in [1, k]$. Moreover, let v_i denote the nodes on which a_i and b_i lie, right after A and B have served the last request of the sequence $\sigma[1, j-1] \equiv \pi(\sigma)[1, j-1]$, and let c_i denote the cost of the edge to which v_i is incident. We can assume, without loss of generality, that A serves request σ_j by moving the server from position a_1 , whereas B serves request $\pi(\sigma_j) \equiv \sigma_j$ by moving server b_2 (hence $c_1 \geq c_2$). We emphasize that indices “1” and “2” will be used throughout the proof to identify these specific servers (a_1, b_1, a_2, b_2) as well as the nodes v_1, v_2 , defined as above.

Given a request r to some node of the star (other than the center), we define \bar{r} as follows:

$$\bar{r} = \begin{cases} a_1, & \text{if } r = a_2 \\ a_2, & \text{if } r = a_1 \\ r, & \text{otherwise.} \end{cases}$$

As a next step, we need to define an appropriate $\pi(\sigma)$ as well as the actions of the algorithm B (relative to the decisions on A on σ). We already stipulated that $\pi(\sigma)[1, j] = \sigma[1, j]$, and we have also determined the decisions of B on the first j requests in $\pi(\sigma)[1, j] = \sigma[1, j]$. We will next define, in an inductive manner, both the bijection, as well as the decisions of B , for all sequences in $\pi(\sigma)[j+1, n]$. For all $\sigma_l \geq j+1$, define $\pi(\sigma_l)$ as follows:

$$\pi(\sigma_l) = \begin{cases} \sigma_l, & \text{A and B have the same configuration after serving } \sigma[1, l-1] \\ & \text{and } \pi(\sigma)[1, l-1], \text{ respectively,} \\ \bar{\sigma}_l, & \text{otherwise.} \end{cases}$$

The mapping $\pi(\sigma)$ is then defined, in the natural way, as $\pi(\sigma_1) \dots \pi(\sigma_n)$. It is straightforward to verify that this mapping is indeed bijective.

We will next inductively ($l \geq j+1$) define how algorithm B serves request $\pi(\sigma_l)$. We first introduce some useful notation. Suppose that after serving sequences $\sigma[1, l]$ and $\pi(\sigma)[1, l]$, respectively, A and B have servers at the same node x . If a_q denotes the server of A that is located on x , then we define $b_{q'}$ to be B 's server that is located on x .

We distinguish the following cases, in order to properly define B .

- If A and B have identical configurations right before serving σ_l and $\pi(\sigma_l)$, respectively, then both A and B serve their requests identically (and thus are in the same configurations right after serving the corresponding requests).
- If A and B do not have identical configurations right before serving σ_l and $\pi(\sigma_l)$, then we consider the following subcases:
 - **Case 1:** If $\sigma_l \in \{v_1, v_2\}$ and server a_2 is at the node of request $\bar{\sigma}_l = \pi(\sigma_l)$, then:
 - * **subcase 1a:** If A serves σ_l by moving server a_2 , then B serves $\pi(\sigma_l)$ by moving b_1 .
 - * **subcase 1b:** If A serves σ_l by moving server a_q with $q \neq 2$, then B serves $\pi(\sigma_l)$ by moving $b_{q'}$. See Figure 2a for an illustration.

- **Case 2:** If $\sigma_l \in \{v_1, v_2\}$ and a_2 is at the node of request σ_l then B serves $\pi(\sigma_l)$ by moving b_1 .
- **Case 3:** If $\sigma_l \notin \{v_1, v_2\}$ then we consider the following subcases:
 - * **subcase 3a:** If A serves σ_l by moving server a_q , with $q \neq 2$, then B moves server $b_{q'}$.
 - * **subcase 3b:** If A serves σ_l by moving server a_2 , then B serves $\pi(\sigma_l)$ by moving b_1 . See Figure 2b for an illustration.

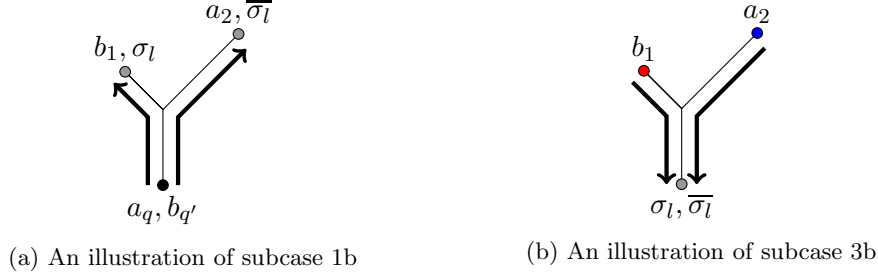


Figure 2: An illustration of subcases 1b and 3a in the statement of algorithm B

The following invariant will be instrumental in proving that algorithm B (as shown above) is well-defined, and that B is bijectively no worse than A .

Lemma 7 (Invariant). *(i) For all requests $\sigma_l, \pi(\sigma_l)$, if A and B are not in the same configuration right before serving these requests, respectively, then either $(a_1, b_2) = (v_1, v_2)$, or $(a_1, b_2) = (v_2, v_1)$.*

(ii) Prior to serving $\sigma_l, \pi(\sigma_l)$, either A and B (respectively) are in identical configurations, or their configurations only differ in that b_1 is not at a node occupied by a server of A , and, likewise, a_2 is not at a node occupied by a server of B .

Proof. The proof is by induction on l . Suppose that the invariant holds right before A and B serve requests σ_l and $\pi(\sigma_l)$, respectively. We will show that the invariant holds after these requests are served by verifying that all cases in the statement of B satisfy the invariant. For succinctness, we will use the expression “before/after the requests” to refer to “immediately before/after serving the corresponding requests”.

- If A and B have identical configurations before the requests, then so they do after the requests, and the invariant holds trivially.
- If A and B do not have identical configurations before the requests, we consider the corresponding cases and subcases of algorithm B .
 - **subcase 1a.** This subcase maintains the invariant because if, say $(a_1, b_2) \in (v_1, v_2)$ prior to the requests, then $(a_1, b_2) \in (v_2, v_1)$ after the requests. Similarly if $(a_1, b_2) \in (v_2, v_1)$ before the requests, then $(a_1, b_2) \in (v_1, v_2)$ after the requests.
 - **subcase 1b:** After the requests, A and B are in the same configuration, so the invariant is maintained (see also Figure 2).

- **Case 2:** This case trivially maintains the invariant since A and B do not move any servers.
- **subcase 3a:** Part (i) of the invariant holds trivially. Invariant (ii) is maintained because A and B move servers from the same node (say x) to the same node (say y), with $x, y \notin \{v_1, v_2\}$.
- **subcase 3b:** After the requests, A and B are in the same configuration, thus the invariant is maintained.

□

We can now use Lemma 7 (and, in particular, Part (ii)) in order to show that B is well-defined. More precisely, Part (ii) of the lemma implies that B makes well-defined decisions in case 2, as well as subcases 1b, 3a and 3b; all other cases or subcases are trivially well-defined.

Having established the consistency of B , we proceed to the last step of the proof, namely to show that $A(\sigma) \leq B(\pi(\sigma))$. Let δ be equal to $A(\sigma_j) - B(\pi(\sigma_j))$; in words, δ is the difference in the cost incurred by A and B when serving σ_j and $\pi(\sigma_j)$, respectively (recalling the notation we introduced early in this proof, this cost is equal to $\delta = c_1 - c_2$). The following lemma shows, informally, that as long as A and B are in different configurations, A has paid at least δ more than B , and when A and B reach the same configuration, A has paid at least as much as B .

Lemma 8. *Let σ_l and $\pi(\sigma_l)$ denote the current requests that are about to be served by A and B , respectively. Then*

- (i) *If A and B are in the same configuration prior to serving σ_l and $\pi(\sigma_l)$, respectively, then $A(\sigma[1, l]) \geq B(\pi(\sigma[1, l]))$.*
- (ii) *If A and B are not at the same configuration prior to serving σ_l and $\pi(\sigma_l)$, respectively, then $A(\sigma[1, l]) \geq B(\pi(\sigma[1, l])) + \delta$.*

Proof. The proof is by induction on l . Suffices to show that each case in the statement of B maintains the statements (i) and (ii) of the lemma after A and B serve requests σ_l and $\pi(\sigma_l)$, respectively.

If A and B are in the same configuration prior to serving σ_l and $\pi(\sigma_l)$, then A and B serve the requests identically, and thus pay the same cost. Hence the lemma is satisfied trivially. Otherwise, we consider the remaining cases in the statement of B :

- **subcase 1a:** In this case, the servers of A and B move the same distance, thus the lemma holds.
- **subcase 1b:** In this case, A pays on σ_l , in the worst case, a cost δ less than B on $\pi(\sigma_l)$ (see also Figure 2a). Thus the lemma holds.
- **Case 2:** The lemma holds trivially as both A and B pay zero cost (they have each a server at the corresponding request).
- **subcase 3a:** A and B serve their requests at the same cost, thus the lemma holds.
- **subcase 3b:** This case is similar to subcase 1b: namely, A pays on σ_l , in the worst case, a cost δ less than B pays on $\pi(\sigma_l)$.

□

The following corollary completes the proof that A is G -like extendable and shows that the greedy algorithm is optimal. This completes the proof of Theorem 6.

Corollary 9. $A(\sigma) \geq B(\pi(\sigma))$.

Proof. Suppose there is an index l such that, after serving $\sigma[1, l]$ and $\pi(\sigma[1, l])$, A and B are in the same configuration. Then from Lemma 8, $A(\sigma[1, l]) \geq B(\pi(\sigma[1, l]))$. For all subsequent requests in $\sigma[l+1, n]$ and $\pi(\sigma)[l+1, n]$, from the statement of B , we deduce that A and B remain in the same configuration; furthermore, A serves $\sigma(h)$ in the same way as B serves $\pi(\sigma(h))$ (for all $l \leq h \leq n$), and thus $A(\sigma(h)) = B(\pi(\sigma(h)))$. Otherwise, there is no such index l such that, after serving $\sigma[1, l]$ and $\pi(\sigma[1, l])$, A and B are in the same configuration, and Lemma 8 shows that $A(\sigma) \geq B(\pi(\sigma)) + \delta > B(\pi(\sigma))$. In both cases, we obtain that $A(\sigma) \geq B(\pi(\sigma))$. □

3.3 Reordering buffer management

As a third application of our framework, we consider the well-studied *reordering buffer management* problem, introduced by Räcke et al. [38]. It consists of a service station that has some active colour, an initially empty buffer of size k and a sequence of coloured requests. Requests enter the buffer sequentially and all items within the buffer that are the same colour as the active colour can be served by the service station. If none of the items in the buffer have the active colour, the service station must change its active colour at a fixed cost. The goal is to minimize the number of colour switches. As with the other problems we consider in this paper, the reordering buffer management problem has been studied extensively in the context of competitive analysis (see, e.g. [1, 7, 6] and references therein).

For this problem, we define G as the greedy algorithm that switches (only if necessary) to a colour c for which the number of items of colour c in the buffer is maximized among all colours (and is thus trivially greedy-like). We once again rely on Theorem 4 in order to show bijective optimality. The nature of this problem gives rise to some technical complications in the optimality proof, in the sense that an algorithm may delay processing a request, (an option that is not meaningful in the context of paging/ k -server problems), which in turn complicates the comparison of $A(\sigma)$ and $B(\pi(\sigma))$ on a request-by-request manner.

Theorem 10. *GREEDY is bijectively optimal for reordering buffer management.*

For the reordering buffer management problem, the next item in the request sequence to enter the buffer is called the *current request*. It is useful to define some notion of time for this problem. At time step i , the current request is σ_{i+1} . That is, the items $\sigma_1, \dots, \sigma_i$ have entered the buffered (and possibly have been served). More precisely, “at time step i ” refers to the precise moment that σ_i enters the buffer. Moreover, without loss of generality, we assume that the current request enters the buffer as soon as a slot is freed. At time n , all the items have entered the buffer and there is no current request.

For this problem, there is a natural notion of “laziness”, as defined in [38], where the algorithm only changes its active colour when otherwise it can no longer make any progress in the input. Without loss of generality, we can assume that all the algorithms are lazy [38].

With these notions, we get the following observation that will be useful for Theorem 10.

Observation 11. For some $\tau < \tau'$, consider any lazy algorithm A at time τ and any lazy algorithm A' at time τ' for a given request sequence σ . All the items in the buffer of A either have been served by A' or are in the buffer of A' . This implies that, if A switches to a colour c that is not in the buffer of A' at time τ' , A advances to some time $\tau'' \leq \tau'$.

As in [38], a *colour block* is the set of items of the same colour that are served with a single colour switch whereas a *buffer colour block* is the set of items of the same colour in the buffer. For some algorithm A , $\mathcal{Z}_\tau^A(\sigma)$ are the buffer colour blocks at time τ for σ .

Let A denote any online algorithm that is G -like on the suffix $\sigma[j+1, n]$ for some $j \in [1, n]$. From Theorem 4, it suffices to prove that A is G -like extendable on j . We will show the existence of an appropriate online algorithm B and a bijection π , according to Definition 3. In particular, since the definition requires that $\pi(\sigma)[1, j] = \sigma[1, j]$, and since B makes the same decisions as A on $\sigma[1, j-1]$, we only need to define $\pi(\sigma)[j+1, n]$, as well as the decisions of B while serving the latter sequence of requests. If A is G -like on $\sigma[i+1]$, we can define B as A and the claim follows. Hence, for the rest of the proof, we assume that A is not G -like on σ_i .

Let x be the active colour of A after the colour switch at time i and let $y \neq x$ be a colour of maximum cardinality in the buffer of A at time i .

In order to define π , we define a bijective mapping. Given a colour (or a request) r , we define \bar{r} as follows.

$$\bar{r} = \begin{cases} x, & \text{if } r = y \\ y, & \text{if } r = x \\ r, & \text{otherwise.} \end{cases}$$

Given a sequence of requests $\sigma = \langle \sigma_1, \dots, \sigma_n \rangle$, we define $\bar{\sigma} = \langle \bar{\sigma}_1, \dots, \bar{\sigma}_n \rangle$. The bijection π is defined such that all the request after j have the colours x changed to y and y changed to x . More formally, $\pi(\sigma) = \sigma[1, j] \cdot \bar{\sigma}[j+1, n]$.

Now, we will define the actions of B for $\pi(\sigma)$. For the requests $\pi(\sigma)[1, j] = \sigma[1, j]$, Algorithm B performs the same colour changes as A . Note that this ensures that the contents of the buffers of A and B are the same at time j . For request σ_j , B must make a colour switch since A makes a colour switch. Algorithm B switches to colour y , a colour of maximum cardinality in the buffer. Let $\Phi = |Z_y| - |Z_x|$, where $Z_y, Z_x \in \mathcal{Z}_j^A$ are the buffer blocks of colour y and x .

For the remaining requests, B will simulate A on σ and maintain a queue of the colour switches of A after time j , where a colour switch to colour c by A is enqueued as \bar{c} . Whenever B must make a colour switch, it removes colours from the queue until the dequeued colour matches the colour of a request in the buffer. This ensures that B will not make more colour switches than A . In the following lemma, we show that B is well-defined and serves all the remaining requests.

Lemma 12. Algorithm B is well-defined over $\pi(\sigma)$.

Proof. Algorithm B is well-defined on $\pi(\sigma)[1, j] = \sigma[1, j]$ as it performs the same actions as A does over $\sigma[1, j-1]$ and then switches to a colour in the buffer at σ_j .

Now, consider the request sequence $\pi(\sigma[j+1, n])$. Let Q be the sequence of colours that B dequeues from its queue over $\pi(\sigma)$, i.e., the list of colour switches of A after time j . We will show, by induction on the indexes of Q , that B is well-defined and will be able to serve the remaining requests of $\pi(\sigma)$. Specifically, we will show that the following invariants are maintained throughout.

For a dequeued colour c , let τ' be the time of B when c is dequeued and let τ be the time step of the colour switch in A . Let Y_τ^{ALG} (resp. X_τ^{ALG}) be the items of colour y (resp. x) in the buffer at

time τ with an index more than j . The following invariants imply the correctness and completeness of B :

1. $\tau' \geq \tau$;
2. for every $Z \in \mathcal{Z}_\tau^A(\sigma)$, if there exists a $Z' \in \mathcal{Z}_{\tau'}^B(\pi(\sigma))$ of the same colour, then $|Z| \leq |Z'|$ and, for every $\sigma_q \in Z$, $\pi(\sigma)_q \in Z'$; and
3. $|Y_\tau^A| \leq |X_{\tau'}^B|$ and $|X_\tau^A| \leq |Y_{\tau'}^B|$ and, for every $\sigma_q \in Y_\tau^A \cup X_\tau^A$, $\pi(\sigma)_q \in X_{\tau'}^B \cup Y_{\tau'}^B$.

Prior to the colour switch at time $i + 1$, the buffers of A and B are identical. After serving X^A , Algorithm A is at time τ and, after serving Y^B , Algorithm B is at time $\tau' \geq \tau + \Phi$. Moreover, this means that B will read more requests from the input than A and, hence, the second and third invariants hold.

Assume that the invariants hold from index $1, \dots, \ell - 1$ and consider the ℓ -th element of Q . There are three cases to consider: (1) there are no items of colour $Q[\ell]$ in the buffer of B at time τ' , (2) there are items of colour $Q[\ell] \neq x$ in the buffer of B at time τ' , and (3) there are items of colour $Q[\ell] = x$ in the buffer of B at time τ' .

- **Case 1:** There are no requests of colour $Q[\ell]$ in the buffer of B at time τ' .
 - **Invariant 1:** By Observation 11¹, after the colour switch of $Q[\ell]$, A is at time $\tau'' \leq \tau'$.
 - **Invariant 2:** Assume for contradiction that items of a colour c enter the buffer of A during this colour switch such that, for $Z_c \in \mathcal{Z}_\tau^A(\sigma)$ and $Z'_c \in \mathcal{Z}_{\tau'}^B(\pi(\sigma))$, $|Z_c| > |Z'_c|$. This can only occur if there exists an $\sigma_j \in Z_c$ with an index greater than all the requests in Z'_c , but that would contradict the fact that $\tau'' \leq \tau'$.
 - **Invariant 3:** By a similar argument, after time i , every request that enters the buffer of A with colour y (resp. x) must be in the buffer of B (but with colour $\bar{y} = x$ (resp. $\bar{x} = y$), maintaining Invariant 3.
- **Case 2:** There are requests of colour $Q[\ell] \neq x$ in the buffer of B at time τ' . By Invariant 2, for $Z_{Q[\ell]} \in \mathcal{Z}_\tau^A(\sigma)$ and $Z'_{Q[\ell]} \in \mathcal{Z}_{\tau'}^B(\pi(\sigma))$, $|Z'_{Q[\ell]}| \geq |Z_{Q[\ell]}|$, and B will make at least as much progress in the request sequence as A . This guarantees all three invariants. The same argument holds for $Q[\ell] = y$ from Invariant 3.
- **Case 3:**
 - There are requests of colour $Q[\ell] = x$ in the buffer of B at time τ' . If this is the first colour switch to x after i , A will read at most Φ requests that are not of colour y that have already been read by B , taking A to time $\tau'' \leq \tau'$. At τ'' , there is additional space for $|Y_\tau^A|$ items not of colour y in the space of the buffer of A . By Invariant 3, $|Y_\tau^A|$ is no greater than the size of $X_{\tau'}^B$ in the buffer of B . The progress made by A after time τ'' is no greater than the progress made by B after time τ' , guaranteeing all three invariants.
 - If this is not the first colour switch to x after i , by Invariant 3, $|Y_\tau^A|$ is no greater than the size of $X_{\tau'}^B$ in the buffer of B . Again, the progress made by A is no greater than the progress made by B , guaranteeing all three invariants.

¹Note that, at this point, the sequences σ and $\pi(\sigma)$ are the same modulo a relabelling of the colour x to y and y to x .

□

Proof of Theorem 10. From Lemma 12, the algorithm B is well-defined. By the definition of B , the number of colour switches of B over $\pi(\sigma)[1, j]$ is the same as of A over $\sigma[1, j]$ and the number of colours switches of B over $\pi(\sigma)[j + 1, n]$ is no more than A over $\sigma[j + 1, n]$. Hence, $B(\pi(\sigma)) \leq A(\sigma)$. □

4 The bijective ratio of the k -server problem on the line and the circle

4.1 Lower bounds

While Theorem 5 shows that GREEDY is bijectively optimal for 2-server on the circle, a similar statement does not hold for the case of the line metric. In fact, in Theorem 13 we prove a stronger statement, namely, we design an explicit online algorithm A which has a lower average-cost ratio against GREEDY of at most c for some constant $c < 1$.

Theorem 13. *For 2-server on the line, and requests over \mathcal{I}_n , there is an online algorithm A and constants c, c' with $c < 1$, such that $\sum_{\sigma \in \mathcal{I}_n} A(\sigma) \leq c \sum_{\sigma \in \mathcal{I}_n} \text{GREEDY}(\sigma) + c'$.*

Proof. Suffices to show that, if we chose a sequence $\sigma \in \mathcal{I}_n$ uniformly at random, then $\mathbb{E}(A(\sigma)) \leq c \cdot \mathbb{E}(\text{GREEDY}(\sigma)) + c'$. Let $x_1, x_2 \in [0, 1]$ with $x_1 < x_2$ be the two server positions of a configuration C . We say that C is *unfavourable* for GREEDY if $x_1 \leq t/3$ and $(2/3)t \leq x_2 \leq t$ with $t \in (0, 1)$ a small constant that we will choose later.

We prove the theorem in two steps. The first step shows that there is an algorithm that essentially simulates GREEDY, but, when faced with an unfavourable situation for GREEDY, it can outperform GREEDY by a constant factor. The second step is then to show that after starting in an arbitrary configuration, GREEDY will find itself in an unfavourable configuration within a constant number of requests with some probability $p > 0$.

We now formally define Step 1 and Step 2.

Step 1. Here we show that there is a $c_1 < 1$ for which the following holds. If C is a configuration that is unfavourable for GREEDY then there is an algorithm A such that for request sequences of length exactly 3 we have $\mathbb{E}(A) \leq c_1 \cdot \mathbb{E}(\text{GREEDY})$ and the final configuration after the three requests are processed by A or by GREEDY is the same.

Step 2. Here we show that, for every fixed $t \in (0, 1)$, starting from an arbitrary configuration (y_1, y_2) , GREEDY will find itself in an unfavourable configuration (x_1, x_2) with $x_2 < t$ within a constant number of steps (depending on t) with positive probability $p > 0$ (also depending on t).

We now proceed with the details in the analysis of the two steps.

Analysis of Step 1: Algorithm A works as follows. Let $\sigma_1, \sigma_2, \sigma_3$ be three requests. Define the positions

$$x_3 = (5/8)x_2 + (3/8)x_1, \text{ and} \tag{3}$$

$$x_4 = 10x_2 \tag{4}$$

(see Figure 3). On the first request, Algorithm A uses Server 1 (at that time positioned at x_1) to serve σ_1 if $(1/2)x_1 + (1/2)x_2 \leq \sigma_1 \leq x_3$. In this case we say A was successful (at outperforming

GREEDY) for the first request. If σ_1 does not satisfy these inequalities then Algorithm A simply simulates GREEDY on all requests.

Figure 3: The figure depicts an unfavourable configuration for GREEDY and the intervals within which the three subsequent requests need to appear so that the new algorithm outperforms GREEDY.



On the second request, if A was successful for σ_1 , then the algorithm serves request σ_2 using Server 2 whenever $x_4 \leq \sigma_2$. We say that A was successful on the second request. In every other case, Algorithm A simulates GREEDY. For that purpose, it places both servers to the positions they would be in if GREEDY had been executed on $\sigma_1\sigma_2$ to begin with.

On the third request, Algorithm A simply simulates GREEDY, again meaning that it places both servers to the positions in which they would be if GREEDY had been executed on $\sigma_1\sigma_2\sigma_3$.

We will now compare the costs of A and the GREEDY algorithm. If A is unsuccessful in the first step, then A and GREEDY have the same costs. The probability that A is successful in the first step is at least $x_3 - (x_2/2 + x_1/2) > k$ for some constant $k > 0$. Assuming thus that A is successful on the first request, A incurs a cost that is at most $d(x_3, x_1) - d(x_2, x_1)$ larger than the cost of GREEDY. Denoting by D_i the average of the cost of A minus the cost of GREEDY in step $i \in \{1, 2, 3\}$, we obtain

$$D_1 \leq k \cdot (d(x_3, x_1) - d(x_2, x_1)).$$

We compare the cost of the second step under the assumption that A was successful in the first step. If A is successful in the second step, then A incurs a cost that is at least $d(x_2, x_3)$ smaller than the one of GREEDY. This happens with probability $(1 - x_4)$. If A is unsuccessful, then A has a cost that is at most $d(x_2, x_1)$ larger than the one of GREEDY. We obtain an average difference in cost of

$$D_2 \leq k \cdot (x_4 \cdot d(x_2, x_3) - (1 - x_4) \cdot d(x_2, x_3)).$$

We now compare the costs for the third request. If A was unsuccessful in one of the previous steps, then the cost of A on the third request is the same as that of GREEDY. Otherwise, we consider two cases. If $x_2 \leq \sigma_3 \leq (\sigma_2 - x_1)/2$, which happens with probability at least $(x_4 - x_2)/2$, then GREEDY serves σ_3 by moving Server 1. In comparison to the cost of GREEDY, algorithm A saves at least $d(x_1, x_2)/2$. Otherwise, if σ_3 is outside of said range, the cost of A is at most $d(x_3, x_1)$ greater than the cost of GREEDY because the total difference of the two configurations is at most $d(x_3, x_1)$. This happens with probability at most x_4 . We obtain

$$D_3 \leq k \cdot (1 - x_4) \cdot (x_4 d(x_3, x_1) - (x_4 - x_2)/2 \cdot d(x_1, x_2)/2).$$

Substituting x_3 and x_4 , using (3) and (4) respectively, overall we obtain that $D_1 + D_2 + D_3 \leq -(1/2)k(x_2 - x_1)x_2(23 - 80x_2)$. Thus, if t (and hence x_2) is sufficiently small, then, on average, the cost of algorithm A is smaller by a constant amount than the cost of GREEDY.

Analysis of Step 2: Let t be a fixed number in $(0, 1)$. Starting from an arbitrary configuration C with constant probability after the first request, server two is in the right half of the line. Assuming this, after the second request, with probability at least $t/3$, Server 1 is located at a position $x_1 \leq t/3$.

On subsequent requests as long as Server 2 is located to the right of t , there is a positive probability that Server 2 moves a distance of at least $(1/6)t$ towards Server 1 but not beyond $(2/3)t$. Thus, with positive probability after at most $6/t$ steps Server 2 is at a position with $(2/3)t \leq x_2 \leq t$ and Server 1 has not moved. \square

Next, we will show a lower bound on the bijective ratio of *any* lazy, deterministic online algorithm for the 2-server problem on the line (which also extends to general k -server on both the circle and the line). The following proposition is useful in establishing lower bounds on the bijective ratio of a given online algorithm A against an algorithm B that may be online or offline. Its proof follows from the fact that under any bijection π , there is at least one sequence σ such that $B(\pi(\sigma)) \leq c$ and $A(\sigma) \geq \rho c$.

Proposition 14. *Suppose that there are $c > 0$ and n_0 such that, for all $n \geq n_0$, $|\{\sigma \in \mathcal{I}_n : A(\sigma) < \rho \cdot c\}| < |\{\sigma \in \mathcal{I}_n : B(\sigma) \leq c\}|$. Then the bijective ratio of A against B is at least ρ .*

Theorem 15. *Any deterministic online algorithm for 2-server even on a line metric of three equidistant points has strict offline bijective ratio at least 2.*

Proof. We will show that there exists an initial server configuration which yields the bound. Suppose that the three points on the line metric are numbered 1,2,3, from left to right, and that the initial server positions are at the end points of the line, i.e., points 1 and 3. Let also d denote the distance between two consecutive points on the line. For a given $n \geq n_0$, define the set of sequences $S \subseteq \mathcal{I}_n$ such that $\sigma[i] \in \{1, 3\}$, for all $i \leq n-2$, $\sigma[n-1] \in \{2\}$, and $\sigma[n] \in \{1, 3\}$. The optimal offline algorithm can serve every sequence in S at a cost equal to d . In contrast, there exists a $\sigma \in S$ such that any deterministic online algorithm A must pay at least $2d$ to serve σ . Namely, if A serves the request $\sigma[n-1]$ by moving server 1, the sequence σ with $\sigma[n] \in \{1\}$ has this property (symmetrically if A serves the request $\sigma[n-1]$ by moving server 3). Last, note that d is the cheapest non-zero cost at which a sequence in \mathcal{I}_n can be served. We thus obtain that $|\{\sigma \in \mathcal{I}_n : A(\sigma) < 2 \cdot d\}| < |\{\sigma \in \mathcal{I}_n : \text{OPT}(\sigma) \leq d\}|$, and the theorem follows from Proposition 14. \square

We note that the lower bound of Theorem 15 extends to general k -server on both the circle and the line.

Last, in the following theorem, we show a lower bound on GREEDY under the Max/Max ratio for the line which implies a lower bound on the bijective ratio.

Theorem 16. *For any $\varepsilon > 0$, the bijective ratio of GREEDY is at least $\frac{k}{2} - \varepsilon$ for the line and at least $\frac{k}{3} - \varepsilon$ for the circle.*

Proof. We show the result for the line. (The bound for the circle follows with a similar argument.) Consider a line that runs from 0 at the left-most point to 1 at the right-most point. Let the servers be labelled s_1, s_2, \dots, s_k from left to right (in any configuration, including the initial). First, we show that it is possible to force GREEDY to move all the servers close to 0 in a constant number of requests. Specifically, define $\delta := \varepsilon/k$. For any constant $\delta', \frac{2\delta}{k-1} \geq \delta' > 0$, we force GREEDY to move s_1 to 0 and to move all the other servers towards 0 so that $d(s_i, s_{i+1}) < \delta'$. An initial request is placed at 0, forcing GREEDY to move s_1 to 0. Iterating on i from 2 to $k-1$, requests are placed as close as possible to the mid-point between s_{i-1} and s_i so that GREEDY moves s_i , continuing until $d(s_{i-1}, s_i) < \delta'$. In total, this requires at most $\lceil \log(1/\delta') \rceil (k-2) + 1$ requests.

The remaining requests alternate between $x := \frac{1}{2} + \frac{k-1}{2}\delta'$ and 1. Note that x is past the mid-point of 1 and s_{k-1} . Hence, these requests (except possibly the first one) will be served by s_k at a cost of more than $1/2 - \delta = 1/2 - \varepsilon/k$.

The worst-case for K-CENTER occurs when all the requests are to an extreme point from any server. This has a total cost of $1/k$ ($1/(2k)$ to serve the request and $1/(2k)$ to return). For large enough n , this gives a Max/Max ratio of $k/2 - \varepsilon$ which implies the theorem.

By a similar argument, it is possible to force a server of GREEDY to travel at least $1/3 - \varepsilon/k$ on the circle for all but a constant number of requests. \square

4.2 Upper bounds

We now consider upper bounds and provide sufficient conditions for showing that the bijective ratio of an online algorithm A against an algorithm B (that may be online or offline) is at most c ; these conditions are formally described in Lemma 17 and Lemma 18. Both lemmas require two conditions stated in terms of individual requests, and their combination yields the desired bound. We use the notation $A(\sigma[i]|B(\sigma[1, i-1]))$ to denote the cost of A for serving request $\sigma[i]$ assuming a configuration resulting from B serving sequence $\sigma[1, i-1]$, where the suffix of the request sequence is implied. For the k -server problem in particular, and the algorithms we consider, this is a well-defined concept. Since B can be either offline or online, we further emphasize that, in the former case, it is implicit that the decisions of B on $\sigma[1, j]$ are contingent on its acting on the entire sequence $\sigma[1, n]$. Last, we use $A(\sigma[i])$ to denote $A(\sigma[i]|A(\sigma[1, i-1]))$.

Lemma 17. *Suppose that there exists a $c > 1$, a $d > 0$ and a bijection π over \mathcal{I}_n such that, given an online algorithm A and an algorithm B , for all $\sigma \in \mathcal{I}_n$ and all $i \leq n$, the following hold:*

- (i) $A(\sigma[i]|B(\sigma[1, i-1])) \leq d \cdot B(\sigma[i])$, and
- (ii) $A(\sigma[i]) - A(\pi(\sigma)[i]|B(\pi(\sigma)[1, i-1])) \leq (c - d) \cdot B(\pi(\sigma)[i])$,

then, $A(\sigma) \leq c \cdot B(\pi(\sigma))$.

Proof. Using $\pi(\sigma)$ as the request sequence for (i) and adding both inequalities, we get $A(\sigma[i]) \leq c \cdot B(\pi(\sigma)[i])$. The lemma follows by summing over all the requests. \square

The following lemma formalizes a similar approach using amortized analysis.

Lemma 18. *Given an online algorithm A and an algorithm B , let Φ be any potential function such that the amortized cost of A for $\sigma[i]$ is $a_i = A(\sigma[i]) + \Delta\Phi_i$, where $\Delta\Phi_i = \Phi_i - \Phi_{i-1}$, and Φ_0 is the potential prior to serving the first request. Suppose also that there exist $c, d > 0$ and a bijection π over \mathcal{I}_n such that, for all $\sigma \in \mathcal{I}_n$ and all $i \leq n$, the following hold:*

- (i) $A(\sigma[i]|B(\sigma[1, i-1])) \leq d \cdot B(\sigma[i])$, and
- (ii) $a_i \leq c \cdot A(\pi(\sigma)[i]|B(\pi(\sigma)[1, i-1]))$,

then, $A(\sigma) \leq c \cdot d \cdot B(\pi(\sigma)) + \Phi_0 - \Phi_n$.

Proof. Summing Inequality (ii) over all the requests gives

$$\sum_{i=1}^n a_i = A(\sigma) + \Phi_n - \Phi_0 \leq c \sum_{i=1}^n A(\pi(\sigma)[i]|B(\pi(\sigma)[1, i-1])) .$$

Thus,

$$A(\sigma) \leq c \left(\sum_{i=1}^n A(\pi(\sigma)[i] | B(\pi(\sigma)[1, i-1])) \right) + \Phi_0 - \Phi_n \leq cd \cdot B(\pi(\sigma)) + \Phi_0 - \Phi_n ,$$

where the last inequality follows from (i), using $\pi(\sigma)$ as the request sequence. \square

4.2.1 Defining the bijection

This section addresses the definition of a suitable bijection. Let $\delta > 0$ be a positive value, representing the distance between any two adjacent points in the metric space. For a given server configuration C , let \mathcal{P}_C^δ be the sequence of points in the metric space, ordered by their distance from the closest server in C . That is, for all $i \leq |\mathcal{P}_C^\delta|$, $D^{\min}(\mathcal{P}_C^\delta[i]) \leq D^{\min}(\mathcal{P}_C^\delta[i+1])$, where $D^{\min}(\mathcal{P}_C^\delta[i])$ is the distance from $\mathcal{P}_C^\delta[i]$ to the nearest server in configuration C .

Definition 19. For the server configurations C_1 and C_2 , an ordered bijection (OB) is a bijection such that, for all i , $\mathcal{P}_{C_1}^\delta[i]$ is matched to $\mathcal{P}_{C_2}^\delta[i]$. Let A and B be two algorithms; we define $\pi^{A,B}(\sigma) = \pi^{A,B}(\sigma[1] \dots \sigma[n])$ to be any bijection of the form $\pi_1^{A,B}(\sigma[1]) \dots \pi_n^{A,B}(\sigma[n])$ such that $\pi_i^{A,B}$ is the OB of the server configurations of A and B right after serving the sequences $\sigma[1, i]$ and $\pi_1^{A,B}(\sigma[1]) \dots \pi_{i-1}^{A,B}(\sigma[i-1])$, respectively. Note that this definition is applicable to all metric spaces. (When clear from the context, we drop the superscript of π .)

As there may be multiple points at some distance d from the nearest server, each permutation of the points at distance d represents a different bijection. In OB, we assume that ties are broken arbitrarily. We assume for simplicity that both the line and the circle have unit length and that δ is chosen such that there exist points at positions $i/(2k)$, for $i \in [0, 2k]$ (assuming an arbitrary “0” point for the circle). A configuration has the k servers spaced uniformly along the line/circle if there is a server at point $i/(2k)$ for all odd $i \in [0, 2k]$.

We define the *best configuration*² under OB as a configuration C^* such that, for any other configuration C , $D^{\min}(\mathcal{P}_{C^*}^\delta[i]) \leq D^{\min}(\mathcal{P}_C^\delta[i])$ for all points i . The following lemma defines the best configuration for the line and the circle.

Lemma 20. For the line and the circle, a configuration in which the servers are spaced uniformly along the metric space is the best configuration.

Proof. We give the proof for the line (a very similar argument applies for the circle). Let C be the configuration in which all the servers are uniformly spaced along the line. That is, the furthest point on the line from the set of servers is at distance $1/2k$. Note that, under our assumptions about δ such a configuration always exists. By the definition of C , ignoring the first k values of 0, the values of $D^{\min}(\mathcal{P}_C^\delta[i])$ increase by δ every $2k$ steps. More formally,

$$D^{\min}(\mathcal{P}_C^\delta[i]) = \begin{cases} 0, & \text{for } 1 \leq i \leq k \\ \lceil \frac{i-k}{2k} \rceil \cdot \delta, & \text{for } k < i. \end{cases}$$

Let C' be a configuration of the servers that is not C . In the configuration of C' , there must be at least one point on the line with a cost higher than $1/2k$. By the definition of C' , ignoring the first k

²For a given metric, the best configurations may not exist; however, for the circle and the line it does.

values of 0, the values of $D^{\min}(\mathcal{P}_{C'}^\delta)$ increase by δ at most every $2k$ steps (i.e., $D^{\min}(\mathcal{P}_{C'}^\delta[\ell-2k]) + \delta \leq D^{\min}(\mathcal{P}_{C'}^\delta[\ell])$ for all $\ell > 3k$).

Hence, there is a point j such that, for every point $j'' < j$, $D^{\min}(\mathcal{P}_C^\delta[j'']) = D^{\min}(\mathcal{P}_{C'}^\delta[j''])$ and, for every point $j' \geq j$, $D^{\min}(\mathcal{P}_C^\delta[j']) \geq D^{\min}(\mathcal{P}_{C'}^\delta[j'])$. Thus C is a best configuration. \square

Informally, in the following lemma, we compare the “worst” configuration to the best configuration. That is, we bound from above the distances to all points from a server with respect to any two configurations.

Lemma 21. *For any $\delta > 0$ and any two configurations C_1 and C_2 of k servers on the line, $D^{\min}(\mathcal{P}_{C_1}^\delta[i]) \leq 2kD^{\min}(\mathcal{P}_{C_2}^\delta[i])$ for every i .*

Proof. We make the natural assumption that in any configuration, there is no point that is occupied by more than one server. We define W as the *worst configuration*, namely as the one that has the property that, for any configuration C , $D^{\min}(\mathcal{P}_W^\delta[i]) \geq D^{\min}(\mathcal{P}_C^\delta[i])$ for all points i (as for the best configuration, a worst configuration may not necessarily exist for every metric, but we will show that it exists for the line and the circle). For the line, we claim that W corresponds to all the servers being at one of ends of the line. In such a configuration, ignoring the first k points at distance 0, the values of $D^{\min}(\mathcal{P}_W^\delta[i])$ increase by δ at each step. More formally,

$$D^{\min}(\mathcal{P}_W^\delta[i]) = \begin{cases} 0, & \text{for } 1 \leq i \leq k \\ (i - k)\delta, & \text{for } k < i. \end{cases}$$

For every other configuration C , $D^{\min}(\mathcal{P}_W^\delta[i]) \geq D^{\min}(\mathcal{P}_C^\delta[i])$.

Let C^* be the configuration with the servers uniformly spaced along the line. By Lemma 20, C^* is the best configuration.

Consider the ratio of $D^{\min}(\mathcal{P}_W^\delta[i])$ to $D^{\min}(\mathcal{P}_{C^*}^\delta[i])$. Modulo the initial k points at distance 0, the values of $D^{\min}(\mathcal{P}_W^\delta[i])$ increase by δ every step (i.e., $D^{\min}(\mathcal{P}_W^\delta[\ell-1]) + \delta = D^{\min}(\mathcal{P}_W^\delta[\ell])$ for all $\ell > k$) and the values of $\mathcal{P}_{C^*}^\delta$ increase by δ every $2k$ steps (i.e., $D^{\min}(\mathcal{P}_{C^*}^\delta[\ell-2k]) + \delta = D^{\min}(\mathcal{P}_{C^*}^\delta[\ell])$ for all $\ell > 3k$). This ratio is maximized when $i \geq 3k$ and $(i - k) \bmod 2k = 0$, for which it attains a value of $2k$. \square

From Lemma 21 and the notions of the proof, we also obtain the following.

Lemma 22. *For any $\delta > 0$ and any two configurations C_1 and C_2 of k servers on the line or the circle, if $\mathcal{P}_{C_1}^\delta[i]$ is located between two adjacent servers of C_1 , then $D^{\min}(\mathcal{P}_{C_1}^\delta[i]) \leq kD^{\min}(\mathcal{P}_{C_2}^\delta[i])$.*

Proof. From Theorem 21, the best configuration C^* places the servers uniformly along the line and, ignoring the first k values at 0, $D^{\min}(\mathcal{P}_{C^*}^\delta[i])$ increase by δ every $2k$ steps (i.e., $D^{\min}(\mathcal{P}_{C^*}^\delta[\ell-2k]) + \delta = D^{\min}(\mathcal{P}_{C^*}^\delta[\ell])$ for all $\ell > 3k$). Hence, $D^{\min}(\mathcal{P}_{C^*}^\delta[i]) \geq \lceil (i - k)/2k \rceil \delta$ and the claim follows if $D^{\min}(\mathcal{P}_C^\delta[i]) \leq \lceil (i - k)/2 \rceil \delta$, which we show in the following.

For any other configuration C , let $\mathcal{P}_{C(s,t)}^\delta \subset \mathcal{P}_C^\delta$ be the set of points between the servers s and t in configuration C , ordered by the distance to the nearest server. As the points are between two servers, the values of $D^{\min}(\mathcal{P}_{C(s,t)}^\delta[i])$ begin at δ and increase by δ every 2 steps (i.e., $D^{\min}(\mathcal{P}_{C(s,t)}^\delta[1]) = D^{\min}(\mathcal{P}_{C(s,t)}^\delta[2]) = \delta$, and $D^{\min}(\mathcal{P}_{C(s,t)}^\delta[\ell-2]) + \delta = D^{\min}(\mathcal{P}_{C(s,t)}^\delta[\ell])$ for all $\ell > 1$). Hence, as $\mathcal{P}_{C(s,t)}^\delta \subset \mathcal{P}_C^\delta$, it follows that $D^{\min}(\mathcal{P}_C^\delta[i]) \leq \lceil (i - k)/2 \rceil \delta$. \square

4.2.2 Completing the analysis

We will now use the bijection π as defined explicitly in Definition 19, so as to establish our upper bounds on the bijective ratio. An important observation is the following.

Observation 23. *For any $\delta > 0$ and any server configuration C , the cost of GREEDY to serve a request at point $\mathcal{P}_C^\delta[i]$ is $D^{\min}(\mathcal{P}_C^\delta[i])$ and the cost of K-CENTER is $2D^{\min}(\mathcal{P}_C^\delta[i])$.*

From its statement, K-CENTER anchors its servers in the best configuration under OB. By definition, to serve a request, the algorithm moves a server to a request and back to its original position. Hence, we obtain that, for any $\delta > 0$ and the best server configuration C^* , the cost of serving a request for K-CENTER is $2D^{\min}(\mathcal{P}_{C^*}^\delta[i])$. This immediately implies the following:

Theorem 24. *K-CENTER has an asymptotic bijective ratio of at most 2 for the k -server problem on the line and the circle.*

We will now use the framework of Lemma 17, and we begin by applying it to the circle metric. Let B be any online or offline algorithm. First note that by the definitions of GREEDY and K-CENTER, we have the following inequalities.

$$\text{GREEDY}(\sigma[i]|B(\sigma[1, i-1])) \leq B(\sigma[i]) \quad (5)$$

$$2 \cdot \text{GREEDY}(\sigma[i]|K\text{-CENTER}(\sigma[1, i-1])) \leq K\text{-CENTER}(\sigma[i]) \quad (6)$$

Theorem 25. *GREEDY has a bijective ratio of at most k for the k -server problem on the circle. Moreover, GREEDY has a bijective ratio of at most $k/2$ for the k -server problem on the circle against K-CENTER.*

Proof. We begin by proving the first part of the theorem, the second part follows along the same lines.

Part 1: From (5), we have that

$$(k-1)B(\pi(\sigma)[i]) \geq (k-1)\text{GREEDY}(\sigma[i]|B(\sigma[1, i-1])) . \quad (7)$$

In addition, from Lemma 22, it follows that

$$k \cdot \text{GREEDY}(\pi(\sigma)[i]|B(\pi(\sigma)[1, i-1])) \geq \text{GREEDY}(\sigma[i]) \quad (8)$$

since every request on the circle is located between two servers. Adding (7) and (8), we thus obtain that, for any σ ,

$$(k-1)B(\pi(\sigma)[i]) \geq \text{GREEDY}(\sigma[i]) - \text{GREEDY}(\pi(\sigma)[i]|B(\pi(\sigma)[1, i-1])) .$$

Using this with (5) and applying Lemma 17, we obtain the result.

Part 2: Inequality (6) implies

$$(k-1)K\text{-CENTER}(\pi(\sigma)[i]) \geq 2(k-1) \cdot \text{GREEDY}(\sigma[i]|K\text{-CENTER}(\sigma[1, i-1])) . \quad (9)$$

From Lemma 22 it follows that

$$2k \cdot \text{GREEDY}(\sigma[i]|K\text{-CENTER}(\sigma[1, i-1])) \geq 2 \cdot \text{GREEDY}(\sigma[i]) \quad (10)$$

since every request on the circle is located between two servers. Adding (9) and (10), we thus obtain that, for any σ ,

$$(k-1)\text{K-CENTER}(\pi(\sigma)[i]) \geq 2(\text{GREEDY}(\sigma[i]) - \text{GREEDY}(\pi(\sigma)[i]|\text{K-CENTER}(\pi(\sigma)[1, i-1]))) .$$

Using this with (6), and applying Lemma 17, we obtain the result. \square

We now move to the line metric. We first note that by combining (5) and Lemma 21 and by applying Lemma 17, we obtain a *strict* bijective ratio of $2k$ for GREEDY on the line. We will also show a stronger, albeit asymptotic bound of $4k/3$, using amortized analysis based on Lemma 18. First, we show, a general bound for GREEDY on the line as compared to some algorithm B . Later, we will choose appropriately the parameters in the statement of the lemma to compare GREEDY to an arbitrary algorithm and to K-CENTER in particular.

Lemma 26. *Let B be any algorithm for the k -server problem on the line such that $\text{GREEDY}(\sigma[i]|B(\sigma[1, i-1])) \leq dB(\sigma[i])$ for $d > 0$. Suppose that there exist $c_1, c_2 > 0$ such that in the configuration of GREEDY right before σ_i ,*

$$\text{GREEDY}(\sigma[i]) \leq \begin{cases} c_1 \text{GREEDY}(\sigma[i]|B(\pi(\sigma)[1, i-1])), & \text{if } \sigma[i] \text{ is between two servers} \\ c_2 \text{GREEDY}(\sigma[i]|B(\pi(\sigma)[1, i-1])), & \text{otherwise.} \end{cases}$$

Then, for any σ , $\text{GREEDY}(\sigma) \leq d \cdot \frac{2c_2c_1}{c_2+c_1} \cdot B(\pi(\sigma)) + \eta$, where η is a constant that depends on the diameter of the line.

Proof. This proof makes use of a potential function argument. Define α to be equal to $\frac{c_2-c_1}{c_2+c_1}$. We define the potential function $\Phi := -\alpha \sum_{i=1}^{k-1} d(g_i, g_{i-1})$, i.e., $-\alpha$ times the sum of the distances between adjacent servers. Let a_i denote the amortized cost for $\sigma[i]$, i.e., $a_i = \text{GREEDY}(\sigma[i]) + \Delta\Phi_i$, where $\Delta\Phi_i = \Phi_i - \Phi_{i-1}$. We distinguish between the following cases, concerning each request $\sigma[i]$.

- *Case 1: $\sigma[i]$ is between two non-outer-most servers.*

In this case, the change in potential is $\Delta\Phi_i = 0$ as the server that moves approaches one adjacent server by a distance of $B(\sigma[i])$ and moves away from its other adjacent server by the same distance. Hence, we have an amortized cost

$$a_i = \text{GREEDY}(\sigma[i]) \leq c_1 \text{GREEDY}(\sigma[i]|B(\pi(\sigma)[1, i-1])) .$$

- *Case 2: $\sigma[i]$ is between an end-point and an outer-most server.*

In this case, $\Delta\Phi_i = -\alpha \text{GREEDY}(\sigma[i])$ and

$$a_i = (1 - \alpha) \text{GREEDY}(\sigma[i]) \leq c_2(1 - \alpha) \text{GREEDY}(\sigma[i]|B(\pi(\sigma)[1, i-1])) .$$

- *Case 3: $\sigma[i]$ is between an outer-most server and its adjacent server.*

In this case, $\Delta\Phi_i = \alpha \text{GREEDY}(\sigma[i])$ and

$$a_i = (1 + \alpha) \text{GREEDY}(\sigma[i]) \leq c_1(1 + \alpha) \text{GREEDY}(\sigma[i]|B(\pi(\sigma)[1, i-1])) .$$

Overall, for any i , $a_i \leq c_1(1 + \alpha) \text{GREEDY}(\sigma[i]|B(\pi(\sigma)[1, i-1]))$ as $c_2(1 - \alpha) = c_1(1 + \alpha) \geq c_1$. The lemma follows by applying Lemma 18. \square

Using Lemma 26 and the properties of the OB, we obtain the following bounds for GREEDY against any algorithm (including offline), as well as K-CENTER.

Theorem 27. *Let B be any algorithm for the k -server problem on the line. Then for any σ , $\text{GREEDY}(\sigma) \leq \frac{4k}{3}B(\pi(\sigma)) + \eta$, where η is a constant. Moreover, $\text{GREEDY}(\sigma) \leq \frac{2k}{3}\text{K-CENTER}(\pi(\sigma)) + \eta$, where again η is a constant.*

Proof. We determine appropriate values for d , c_1 , and c_2 in the statement of Lemma 26. For the first part of the theorem, $d = 1$ from (5), $c_1 \leq k$ from Lemma 22, and $c_2 \leq 2k$ from Lemma 21; hence the first part of the theorem follows. For the second part of the theorem, c_1 and c_2 are as above, and, from (6), we have $d = 1/2$. \square

5 The bijective ratio of the k -server problem on the star

In this section, we study the bijective ratio of the continuous k -server problem on the star. Here, a star consists of m line segments (called *rays*), not necessarily of the same length, which have a common origin called the *center*. One can think of such a metric as a transitional metric when moving from the line to trees that allows us to draw certain interesting conclusions concerning the performance of algorithms under the bijective ratio. Similar to the line, we represent this metric by a *spider* graph, which consists of a set of paths (of potentially different lengths) that intersect at the center, and in which all edges have the same length.

Recall that on the line, Theorem 24 shows that K-CENTER has a bijective ratio of 2, whereas Theorems 27 and 16 show that GREEDY has a bijective ratio of $\Theta(k)$. In particular, our analysis of the bijective ratio of K-CENTER matches its Max/Max ratio [11]. In contrast to our analysis of the bijective ratio of K-CENTER and GREEDY, we show that on stars the bijective performance of these algorithms changes in a dramatic way. More precisely, in Theorem 28, the bijective ratio of K-CENTER is unbounded, while, in Theorem 30, we show that the bijective ratio of GREEDY is at most $4k$. These results demonstrate that the bijective ratio is not only a generalization of Max/Max ratio, but it also classifies algorithms very differently in terms of performance.

Theorem 28. *There exists a star S , and an online algorithm A such that K-CENTER has unbounded asymptotic bijective ratio against A on S .*

Proof. Consider a star S that consists of $m - 1$ rays of length d and one longer ray of length $4kd - d$. For this star, K-CENTER anchors its servers on the long ray (see Figure 4). More specifically, the first server is placed at a distance d from the center of the star with the remaining servers placed along the long ray with a spacing of $4d$ between them. We also define an algorithm A that anchors a server at the center of S , and the remaining $k - 1$ servers as in Figure 4. Similar to K-CENTER, A serves a request with the closest server, which then returns it to its anchor position. For this star, we show that there exist integers j and n such that the j -th cheapest sequence of K-CENTER (among sequences in \mathcal{I}_n) is at least $\Omega(d)$ times as costly as the j -th cheapest sequence of A .

We assume that the anchor position of the servers for algorithm A is the initial configuration of servers for the two algorithms.

We observe that after the initial anchoring of the servers for A and K-CENTER, each requested point always incurs the same cost for A , and always incurs the same cost for K-CENTER.

For some $\varphi \leq d$ to be determined later, let $S_{\varphi,n}^{\text{K-CENTER}}$ be the sequences of cost at most $2\varphi n$ for K-CENTER. For a given sequence $\sigma \in S_{\varphi,n}^{\text{K-CENTER}}$, let ε be the number of requests that have cost

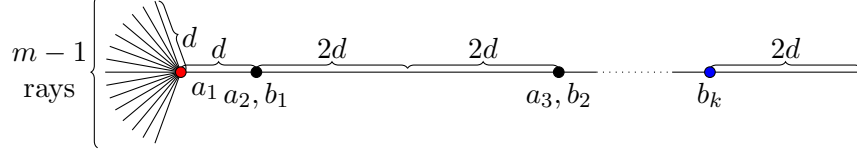


Figure 4: An illustration of the lower bound construction for the K-CENTER algorithm. Here, we denote by a_i , b_i the servers of A and K-CENTER, respectively.

more than $2d$. Hence, $2\epsilon d \leq 2\varphi n \iff \epsilon \leq \frac{\varphi n}{d}$. Based on the anchor points for K-CENTER, there are $2kd + k$ points with cost at most $2d$ and $N - 2kd - k$ point with cost more than $2d$, where N is the total number of nodes in the metric space. We can bound from above the number of sequences in $S_{\varphi, n}^{\text{K-CENTER}}$ as follows. (For this recall that the cost of a request is twice the distance to the nearest server since the server will be moved back to its original position afterwards.) Let

$$\begin{aligned}
|S_{\varphi, n}^{\text{K-CENTER}}| &\leq \sum_{\epsilon=0}^{\frac{\varphi n}{d}} \binom{n}{\epsilon} (2kd + k)^{n-\epsilon} (N - 2kd - k)^{\epsilon} \\
&\leq 2^n \max_{\{0, \frac{\varphi n}{d}\}} \{(2kd + k)^{n-\epsilon} (md + 2kd)^{\epsilon}\} \\
&\leq 2^n (2kd + k)^{n - \frac{\varphi n}{d}} (md + 2kd)^{\frac{\varphi n}{d}}, \text{ for } m > k, \\
&\leq 2^n (3kd)^{n - \frac{\varphi n}{d}} (3md)^{\frac{\varphi n}{d}} \\
&\leq (6kd)^n (3md)^{\frac{\varphi n}{d}}.
\end{aligned} \tag{11}$$

Similarly, for some positive constant $c \ll d$, let $S_{c, n}^A$ be the sequences of cost at most $2cn$ for A . We can bound from below the number of sequences in $S_{c, n}^A$ as follows.

$$|S_{c, n}^A| \geq (c \cdot (m + 2k - 2) + k)^n \geq (cm)^n. \tag{12}$$

Setting $m = (kd)^3$ and $\varphi = d/3$ and using (11), we get that $|S_{d/3, n}^{\text{K-CENTER}}| = 6^n 3^{\frac{n}{3}} m^n$. For $c \geq 9$, $|S_{9, n}^A| > |S_{d/3, n}^{\text{K-CENTER}}|$. Hence, for $j = |S_{d/3, n}^{\text{K-CENTER}}|$, the bijective ratio for the j -th cheapest sequence is at least $d/27$.

This implies that the bijective ratio of K-CENTER against A is at least $\Omega(d)$ and, hence, unbounded. \square

We define our bijection π according to Definition 19. Note that in the star, unlike the line and the circle, a best configuration, as defined in Section 4.2.1 may not necessarily exist. However, the following corollary (which follows from Lemma 21) shows that there exists a configuration that is good enough.

Corollary 29. *Let C be a configuration with a server at the centre. For any $\delta > 0$, $\mathcal{P}_C^\delta[i] \leq 2k\mathcal{P}_{C'}^\delta[i]$, where C' is any other configuration.*

We note that, unlike the line, the approach of Lemma 17 cannot yield a bounded bijective ratio for GREEDY. This is because the best and worst configurations (as defined in Section 4.2.1) can be unbounded with respect to D^{\min} . We will thus resort to amortized analysis, using a different potential function than the one used in Lemma 26.

Theorem 30. *Let B be any algorithm for the k -server problem on the uniform spider graph. For any σ , $\text{GREEDY}(\sigma) \leq 4kB(\pi(\sigma)) + \eta$, where η is a constant.*

Proof. The potential function is $\Phi := \sum_{i=1}^k d(c, g_i)$, where c is the centre of the star and g_i is the i -th server. Let a_i denote the amortized cost for $\sigma[i]$, i.e., $a_i = \text{GREEDY}(\sigma[i]) + \Delta\Phi_i$, where $\Delta\Phi_i = \Phi_i - \Phi_{i-1}$. We distinguish three cases with respect to $\sigma[i]$.

- *Case 1: A server serves $\sigma[i]$ away from the centre on the same ray.* In this case, the change in potential is $\Delta\Phi_i = \text{GREEDY}(\sigma[i])$. Hence, the amortized cost $a_i = 2\text{GREEDY}(\sigma[i]) \leq 2d(c, \sigma[i]) \leq 4k \cdot \text{GREEDY}(\pi(\sigma)[i])B(\pi(\sigma)[1, i-1])$, where the last inequality follows from Corollary 29.
- *Case 2: A server serves $\sigma[i]$ towards the centre on the same ray.* In this case, the change in potential is $\Delta\Phi_i = -\text{GREEDY}(\sigma[i])$. Hence, the amortized cost $a_i = 0$.
- *Case 3: A server serves $\sigma[i]$ by moving a server that lies on a different ray than the one of $\sigma[i]$.* In this case, the change in potential is $\Delta\Phi_i = d(c, \sigma[i]) - d(s_1, c)$. Hence, for the amortized cost we get $a_i = \text{GREEDY}(\sigma[i]) + d(c, \sigma[i]) - d(s_1, c) = 2d(c, \sigma[i]) \leq 4k \cdot \text{GREEDY}(\pi(\sigma)[i])B(\pi(\sigma)[1, i-1])$, where the last inequality follows from Corollary 29.

Thus, $a_i \leq 4k \cdot \text{GREEDY}(\sigma[i])B(\pi(\sigma)[1, i-1])$ for all i . The theorem follows from (5) and Lemma 18. \square

References

- [1] Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. Almost tight bounds for reordering buffer management. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 607–616, 2011.
- [2] Aris Anagnostopoulos, Clément Dombry, Nadine Guillotin-Plantard, Ioannis Kontoyiannis, and Eli Upfal. Stochastic analysis of the k -server problem on the circle. In *Proc. of the 21st International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA)*, 2010.
- [3] Spyros Angelopoulos, Reza Dorrigiv, and Alejandro López-Ortiz. On the separation and equivalence of paging strategies. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 229–237, 2007.
- [4] Spyros Angelopoulos, Reza Dorrigiv, and Alejandro López-Ortiz. List update with locality of reference. In *Proceedings of the 8th Latin American Theoretical Informatics Symposium (LATIN)*, pages 399–410, 2008.
- [5] Spyros Angelopoulos and Pascal Schweitzer. Paging and list update under bijective analysis. *Journal of the ACM*, 60(2), 2013.
- [6] Noa Avigdor-Elgrabli and Yuval Rabani. An improved competitive algorithm for reordering buffer management. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 1–10, 2013.
- [7] Noa Avigdor-Elgrabli and Yuval Rabani. An improved competitive algorithm for reordering buffer management. *ACM Transactions on Algorithms*, 11(4):35, 2015.

- [8] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 2012.
- [9] Yair Bartal and Elias Koutsoupias. On the competitive ratio of the work function algorithm for the k-server problem. *Theor. Comput. Sci.*, 324(2-3):337–345, 2004.
- [10] Alfonso Baumgartner, Robert Manger, and Zeljko Hocenski. Work function algorithm with a moving window for solving the on-line k-server problem. *CIT*, 15(4):325–330, 2007.
- [11] Shai Ben-David and Allan Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994.
- [12] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- [13] Allan Borodin, Morten N. Nielsen, and Charles Rackoff. (incremental) priority algorithms. *Algorithmica*, 37(4):295–326, 2003.
- [14] Joan Boyar, Martin R. Ehmsen, and Kim S. Larsen. Theoretical evidence for the superiority of LRU-2 over LRU for the paging problem. In *Proceedings of the 4th International Workshop on Approximation and Online Algorithms (WAOA)*, pages 95–107, 2006.
- [15] Joan Boyar, Lene M. Favrholdt, and Kim S. Larsen. The relative worst-order ratio applied to paging. *Journal of Computer and System Sciences*, 73(5):818–843, 2007.
- [16] Joan Boyar, Sandy Irani, and Kim S. Larsen. A comparison of performance measures for online algorithms. *Algorithmica*, 72(4):969–994, 2015.
- [17] Joan Boyar, Kim S. Larsen, and Morten N. Nielsen. The accommodating function: A generalization of the competitive ratio. *SIAM J. on Computing*, 31(1):233–258, 2001.
- [18] A. R. Calderbank, E. G. Coffman, Jr., and L. Flatto. Sequencing problems in two-server systems. *Math. Oper. Res.*, 10(4):585–598, 1985.
- [19] A. R. Calderbank, E. G. Coffman, Jr., and L. Flatto. Sequencing two servers on a sphere. *Comm. Statist. Stochastic Models*, 1(1):17–28, 1985.
- [20] Marek Chrobak, Howard J. Karloff, T. H. Payne, and Sundar Vishwanathan. New results on server problems. *SIAM J. Discrete Math.*, 4(2):172–181, 1991.
- [21] Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM J. Comput.*, 20(1):144–148, 1991.
- [22] Marek Chrobak and Lawrence L Larmore. The server problem and on-line games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 7:11–64, 1992.
- [23] Reza Dorrigiv and Alejandro López-Ortiz. A survey of performance measures for on-line algorithms. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 36(3):67–81, September 2005.
- [24] Ronald L Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

- [25] Benjamin Hiller and Tjark Vredeveld. Probabilistic analysis of online bin coloring algorithms via stochastic comparison. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA)*, pages 528–539, 2008.
- [26] Benjamin Hiller and Tjark Vredeveld. Simple optimality proofs for Least Recently Used in the presence of locality of reference. Technical report, Maastricht University of Business and Economics, 2009.
- [27] Benjamin Hiller and Tjark Vredeveld. Probabilistic alternatives for competitive analysis. *Computer Science - R \mathcal{E} D*, 27(3):189–196, 2012.
- [28] Micha Hofri. Should the two-headed disk be greedy? - yes, it should. *Inf. Process. Lett.*, 16(2):83–85, 1983.
- [29] Michael Keane. Interval exchange transformations. *Mathematische Zeitschrift*, 141:25–31, 1975.
- [30] Claire Kenyon. Best-fit bin-packing with random order. In *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 359–364, 1996.
- [31] Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009.
- [32] Elias Koutsoupias and Christos Papadimitriou. Beyond competitive analysis. *SIAM J. on Computing*, 30:300–317, 2000.
- [33] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42(5):971–983, 1995.
- [34] Steven S. Lumetta and Michael Mitzenmacher. Using the power of two choices to improve bloom filters. *Internet Mathematics*, 4(1):17–33, 2007.
- [35] Mark S. Manasse, Lyle A. McGeoch, and Daniel Dominic Sleator. Competitive algorithms for on-line problems. In *STOC*, pages 322–333, 1988.
- [36] Michael Mitzenmacher. Bounds on the greedy routing algorithm for array networks. *J. Comput. Syst. Sci.*, 53(3):317–327, 1996.
- [37] Alfred Mller and Dietrich Stoyan. *Comparison Methods for Stochastic Models and Risks*. Wiley, 2002.
- [38] Harald Räcke, Christian Sohler, and Matthias Westermann. Online scheduling for sorting buffers. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA)*, pages 820–832, 2002.
- [39] Prabhakar Raghavan and Marc Snir. Memory versus randomization in on-line algorithms. *IBM Journal of Research and Development*, 38(6):683–708, 1994.
- [40] Tomislav Rudec, Alfonzo Baumgartner, and Robert Manger. Measuring true performance of the work function algorithm for solving the on-line k-server problem. *CIT*, 18(4), 2010.
- [41] Tomislav Rudec, Alfonzo Baumgartner, and Robert Manger. A fast work function algorithm for solving the k -server problem. *CEJOR*, 21(1):187–205, 2013.

- [42] Sridhar Seshadri and Doron Rotem. The two headed disk: Stochastic dominance of the greedy policy. *Inf. Process. Lett.*, 57(5):273–277, 1996.
- [43] Moshe Shaked and J. George Shanthikumar. *Stochastic Orders and their Applications*. Academic Press, San Diego, 1994.
- [44] Daniel D. Sleator and Robert E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of the ACM*, 28:202–208, 1985.
- [45] Elmar Wolfstetter. *Topics in Microeconomics*. Cambridge University Press, Cambridge, 1999.
- [46] Neal E. Young. On-line caching as cache size varies. In *Proceedings of the Second Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 28-30 January 1991, San Francisco, California.*, pages 241–250, 1991.
- [47] Neal E. Young. The k -server dual and loose competitiveness for paging. *Algorithmica*, 11(6):525–541, 1994.
- [48] Neal E. Young. Bounding the diffuse adversary. In *Proceedings of the 9th Annual ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 420–425, 1998.
- [49] Neal E. Young. On-line paging against adversarially biased random inputs. *Journal of Algorithms*, 37(1):218–235, 2000.
- [50] Neal E. Young. On-line file caching. *Algorithmica*, 33(3):371–383, 2002.